CLAIMS

1. A method for optimizing an executable comprising:
migrating a plurality of objects from a first memory to a second memory;
determining alignment of the migrated plurality of objects; and
eliminating redundant initialization code of the plurality of objects.

2. The method for optimizing an executable of claim 1, wherein the plurality of objects are variables.

3. The method for optimizing an executable of claim 1, the migrating the plurality of objects further comprising:
determining whether each object of the plurality of objects are accessible from a plurality of processors;
determining an equivalence set of aliased objects in the plurality of objects;
determining objects of the plurality of objects eligible for migration;
changing residence of the objects determined to be eligible for migration; and
changing accesses of the objects having their residence changed.

4. The method for optimizing an executable of claim 1, the determining alignment further comprising:
analyzing the migrated objects by forward disjunctive dataflow analysis;
determining a minimum alignment necessary for each migrated object; and
setting the minimum alignment necessary for each migrated object.

5. The method for optimizing an executable of claim 1, wherein the first

memory is an external memory and the second memory comprises a plurality of indexed registers residing in a microengine.

6. A processing device comprising:

an optimizer to migrate a plurality of objects from an external memory of a network processing device to a plurality of registers coupled to a processor, the optimizer further to align and eliminate redundant initialization code of the plurality of objects.

7. The processing device of claim 6, wherein the plurality of registers are indexed.

8. The processing device of claim 6, wherein the plurality of objects are variables.

9. The processing device of claim 6, wherein the migrated plurality of objects are not shared by the processor and at least one other processor.

10. The processing device of claim 6, wherein the network processing device is a router.

11. An optimizer system for network processors comprising:

a processor,

a first memory coupled to the processor;

a display coupled to the processor; and

a compiler to migrate a plurality of objects from a second memory to a plurality of indexed registers in a network processor, the compiler further to align and eliminate redundant initialization code of the plurality of objects.

12. The optimizer system for network processors of claim 11, the compiler

including:

a first determiner to determine whether each object of the plurality of objects are accessible from a plurality of processors in a network device;

a second determiner to determine an equivalence set of aliased objects in the plurality of objects;

a third determiner to determine objects of the plurality of objects eligible for migration;

a migrator to change residence of the objects determined to be eligible for migration; and

an accessor to change accesses of the objects having their residence changed.

13. The optimizer system for network processors of claim 12, wherein the second memory is external to the plurality of processors.

14. The optimizer system for network processors of claim 11, wherein the plurality of objects are variables.

15. The optimizer system for network processors of claim 11, wherein the second memory is external to the plurality of indexed registers.

16. A machine-accessible medium containing instructions that, when executed, cause a machine to:

migrate a plurality of variables from a first memory to a plurality of indexed registers;

align the migrated plurality of variables; and

eliminate redundant initializations to a base address register.

17. The machine accessible medium of claim 16, further comprising instructions that, when executed, cause a machine to:

determine whether each variable of the plurality of variables are accessible from at least two network processors;

determine an equivalence set of aliased variables in the plurality of variables; and

change location of the variables that are determined to be eligible for migration.

18. The machine accessible medium of claim 16, further comprising instructions that, when executed, cause a machine to:

analyze the migrated variables by forward disjunctive dataflow analysis; and

determine a minimum alignment necessary for each migrated variable.

19. The machine accessible medium of claim 16, further comprising instructions that, when executed, cause a machine to:

set the minimum alignment necessary for each migrated variable.

20. The machine accessible medium of claim 16, further comprising instructions that, when executed, cause a machine to:

compile source code to migrate the plurality of variables from the first memory to the plurality of indexed registers.